2025年5月12日

19:45

**课程设计报告要求：**

* 1. 课程设计的题目，尽量从老师统一提供的题目列表中选择，如有小组有自己的特殊想法也可自行确定题目，但需要经过老师审核确认。

* 1. 课程设计报告应严格按照学校要求，必须按顺序依次包含以下内容板块：

|  |
| --- |
| 封面  课程设计任务书  目录  正文  1 系统概述  1.1开发背景和意义 1.2现状描述 1.3系统目标  2 系统分析  2.1系统需求 2.2业务流程分析 2.3数据流程分析 2.4数据字典等  3 系统设计  3.1模块结构设计 3.2数据库设计 3.3物理配置方案设计等  4 系统实现  4.1 开发技术 4.2 开发平台搭建 4.3功能实现  5收获和体会  参考文献（15-30篇国内外文献） |

* 1. 课程设计报告要求每个学生独立编写，并撰写课程设计任务书（附件二），报告不少于**8000**字，封面采用学校统一印制的封面（附件一），并在课程设计任务书说明每个学生的任务分工。
  2. 课程设计分组进行，1-3人一组，自由分组，每组完成同一份设计报告，但每组成员需单独提交设计报告，并在设计报告中说明具体小组的分工内容及小组成员贡献排序。

* 1. 课程设计总成绩评定由设计报告（50%）+系统验收（50%）两部分组成。

（1）设计报告：报告规范、完整，能够正确运用《现代信息系统分析与设计》课程所涉及的专业知识输出准确、专业的报告文档。

（2）系统验收：能够利用本专业综合知识技能（JAVA程序设计、计算机网络、数据库原理与设计等）将报告所设计的系统方案尽可能进行实现和展示。

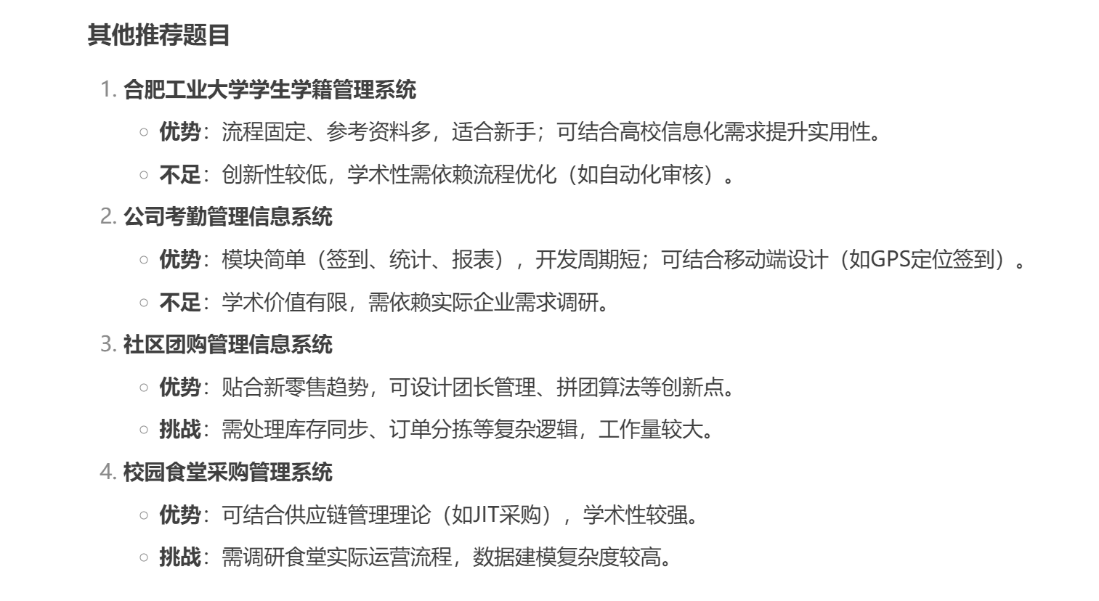
6. 课程设计报告电子版一律以“学号+姓名+报告名称”（例如：“2022800020+张三+图书馆管理系统设计与实现”）统一命名，并将word版+pdf版两个版本按时提交至学习委员处，由学习委员统一提交给老师。

|  |
| --- |
| 课设题目 |
| 合肥工业大学教育超市管理信息系统分析与设计 |
| 锦怡园宾馆管理信息系统分析与设计 |
| 小区物业管理系统分析与设计 |
| 合肥工业大学校医院药品管理系统分析与设计 |
| 合肥长途汽车站营运系统分析与设计 |
| 汽车租赁管理系统分析与设计 |
| 二手房销售管理系统分析与设计 |
| 硕士研究生招生考试网上报名与调剂系统分析与设计 |
| 民航机票管理信息系统分析与设计 |
| 合肥工业大学设备采购管理系统分析与设计 |
| 社区团购管理信息系统分析与设计 |
| 公司考勤管理信息系统分析与设计 |
| 线上校园二手物品交易信息管理系统分析与设计 |
| 企业供应商管理信息系统分析与设计 |
| 校园食堂采购管理信息系统分析与设计 |
| 合肥工业大学学生学籍管理系统分析与设计 |
| 合肥工业大学奖学金评定系统分析与设计 |
| 旅游景点门票管理系统分析与设计 |
| 医院在线医疗预约系统分析与设计 |
| 大学生创新创业项目管理信息系统分析与设计 |
| 管院MBA案例大赛管理信息系统分析与设计 |
| 暑期“三下乡”社会实践信息管理系统分析与设计 |
| 会议室租借管理信息系统分析与设计 |
| 管院优秀大学生夏令营活动管理信息系统分析与设计 |
| 连锁药店药品管理系统分析与设计 |
| 社区居家养老管理信息系统分析与设计 |
| 高校科研项目管理信息系统分析与设计 |

经过一番纠结，最终选择校园二手物品交易信息管理系统（由于相关背景知识基本为0，所以目标是完成任务，而非高端大气上档次）

deepseek和我想法相同





选择项目原则：有源码+部署介绍，光有源码的不看（github上star多没有部署也没法要）

Two days later

最终以gitee的项目为例：https://gitee.com/guyue-personspace/campus-second-hand-items-circulation-platform

时间安排上每天都要接触这个项目，慢慢熟悉，每天了解一点点

部署及介绍视频

<https://www.bilibili.com/video/BV1ou4y1n7Su/?share_source=copy_web&spm_id_from=333.788.comment.all.click&vd_source=bf2f7a70eb189b10102656480ded87c6>

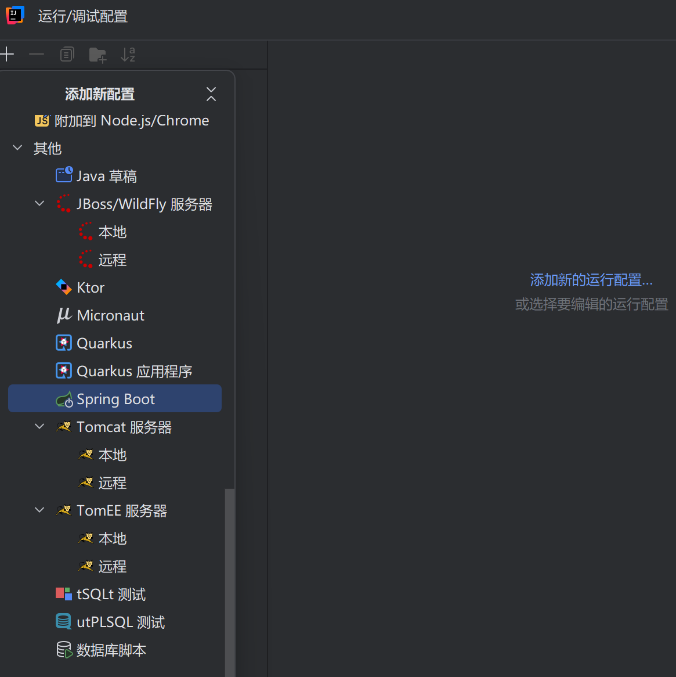
<https://www.bilibili.com/video/BV1Q8411S79i?spm_id_from=333.788.recommend_more_video.-1&vd_source=bf2f7a70eb189b10102656480ded87c6>

工欲善其事，必先利其器

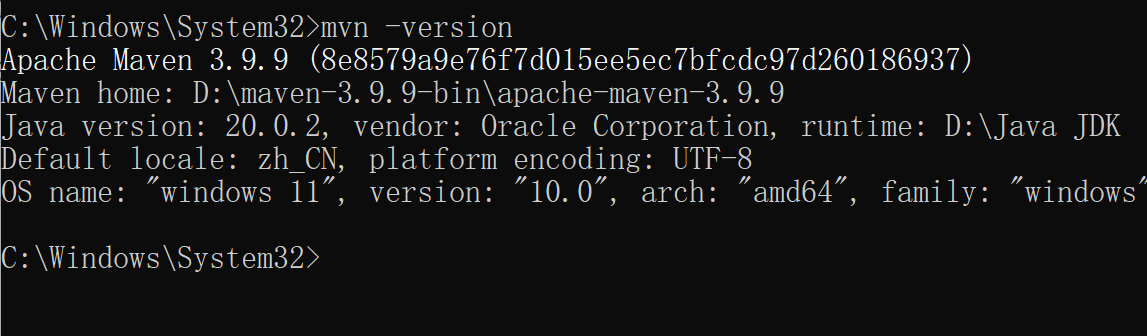
<https://blog.lupf.cn/articles/2024/12/23/1734953685339.html>

为了springboot框架的配置，卸载掉社区版IDEA，使用专业版（感谢互联网）





maven的配置



配置springboot 出现的问题

idea 项目运行出现no spring boot application class specified

<https://blog.csdn.net/weimeibuqieryu/article/details/127525132>

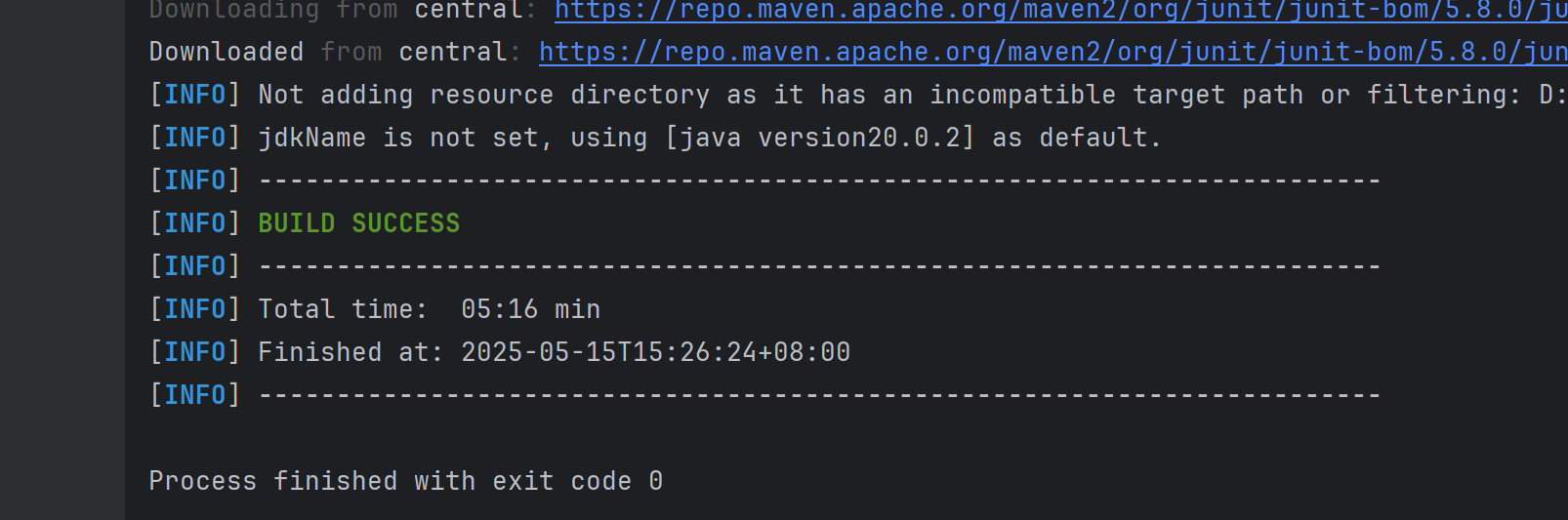
IDEA下载maven半天没效果

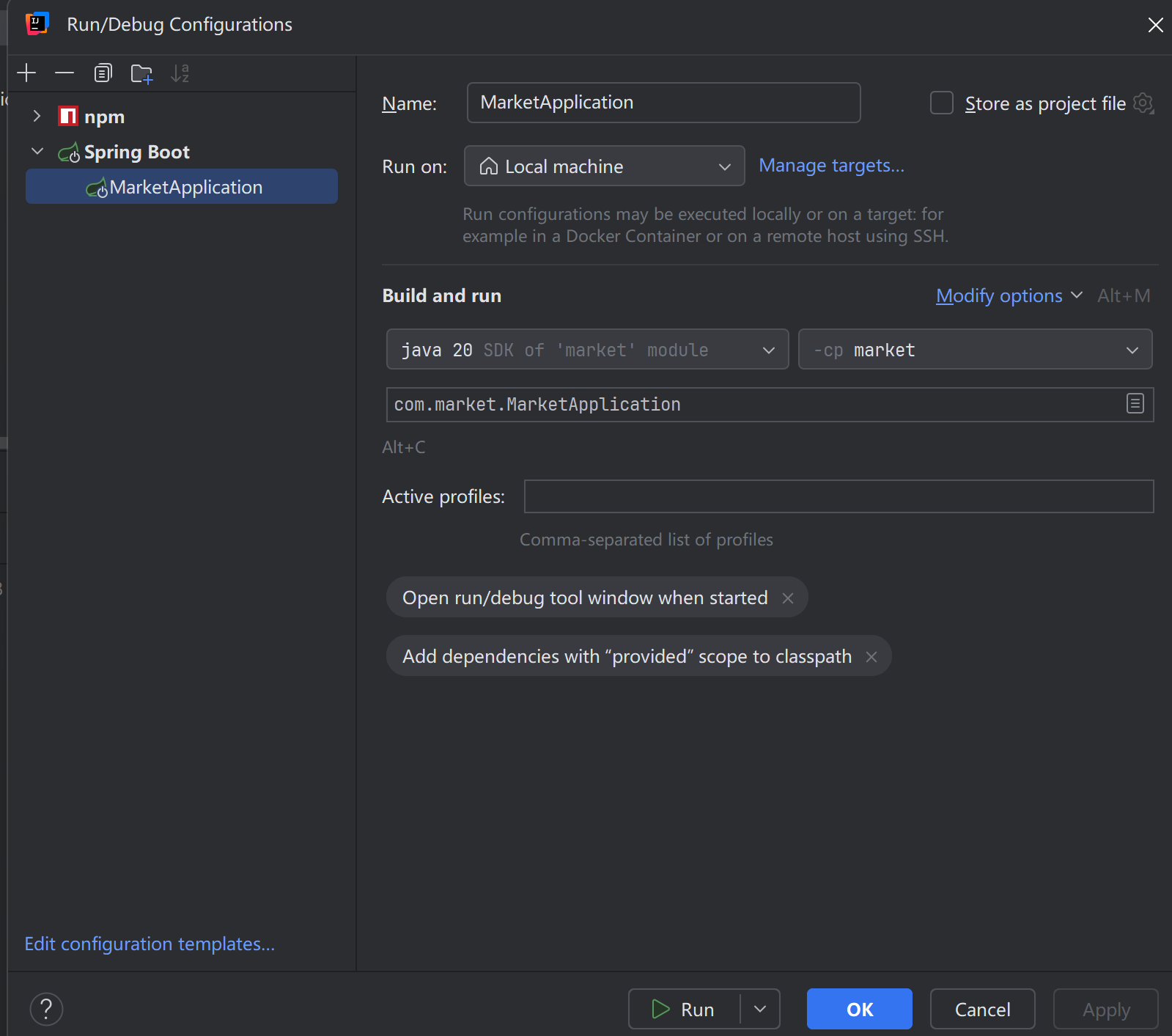
还是得网上自己配置

maven:https://blog.csdn.net/qq\_45867488/article/details/115748607（新项目）

<https://blog.csdn.net/Sareur_1879/article/details/143091933（已有项目）>

<https://blog.csdn.net/weixin_47233778/article/details/110084156> （mvn:idea:idea）





报错解决

<https://blog.csdn.net/WillyWithouty/article/details/116120785>

<https://blog.csdn.net/weixin_39984460/article/details/117825089>

接下来就是前端部分

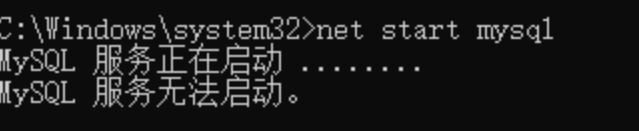
主要先nvm，剩下就好办

<https://www.cnblogs.com/rnny/p/17839190.html#tid-5S87yp>

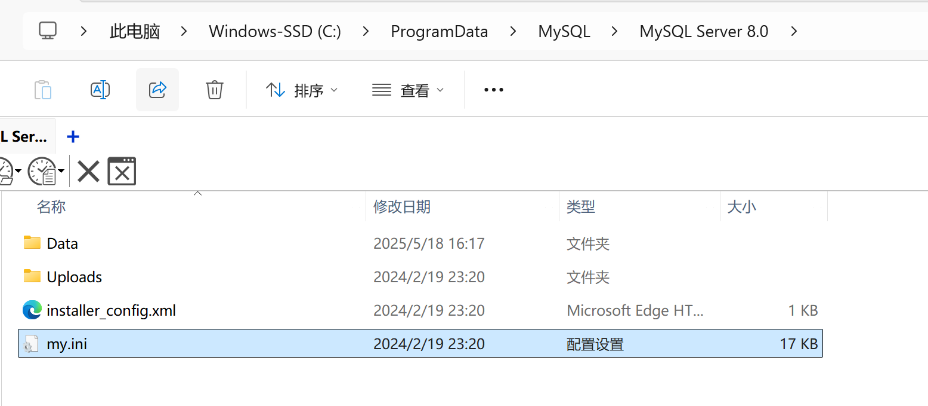
适合小白的 nvm 安装配置教程（Windows版）

之后是数据库部分：https://blog.csdn.net/will\_\_be/article/details/107542003

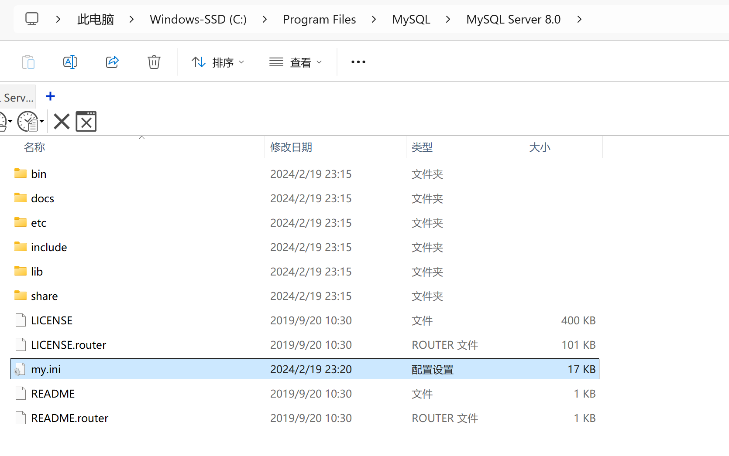
太久没用了，都不知道怎么启动，启动就报错



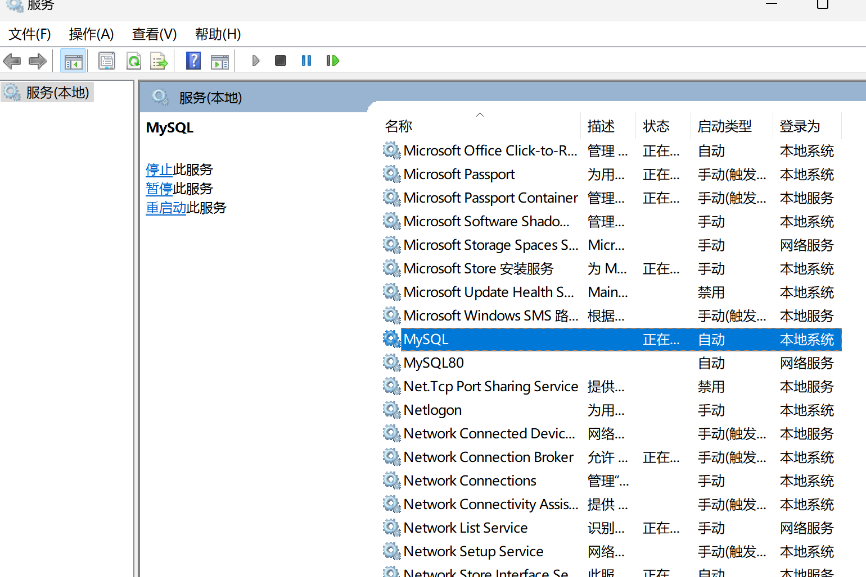
解决关键

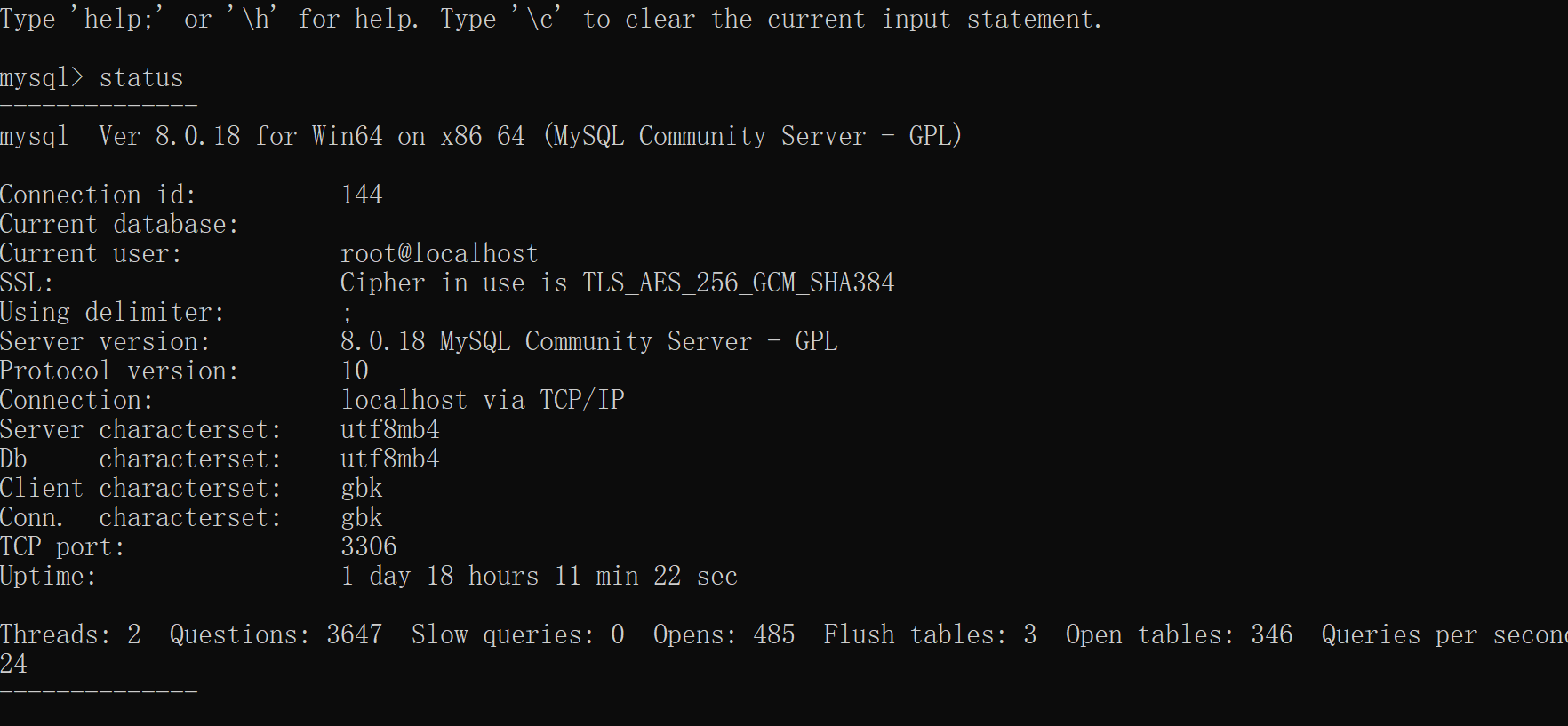


把my.ini配置文件放到bin目录下就成了，差点让我重装，还好重装太麻烦了



之后就成功启动了





navicat就能连接了

遇到问题：账号密码输入后没法登录，推断在后端问题，查看maven的信息

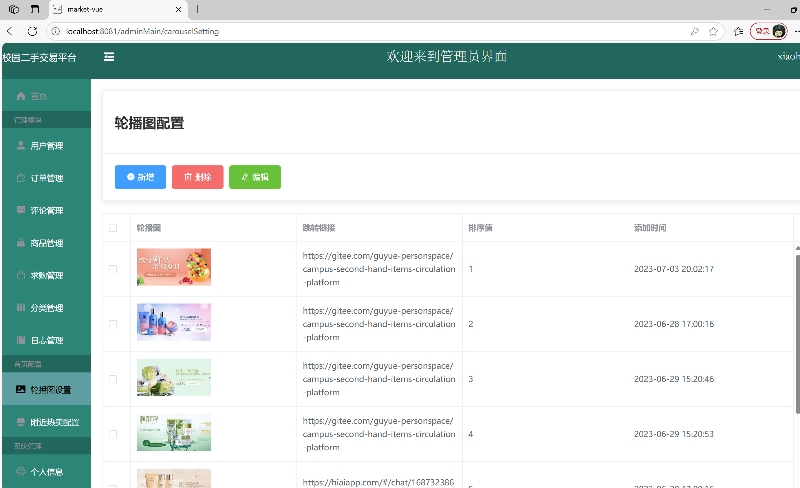
报错：cause: org.springframework.jdbc.CannotGetJdbcConnectionException: Failed to obtain JDBC Connection; nested exception is java.sql.SQLException: Access denied for user 'root'@'localhost' (using password: YES)] with root cause

Springboot JDBC和mysql的连接出现问题

<https://blog.csdn.net/Tec_Xie/article/details/119744110>

总算解决前端登录按钮按了没反应的问题(作为一只小菜鸟，任何一个bug都能让我止步不前)





既然这个项目能部署（感谢互联网开源），接下来就是在此基础上进行二次开发，为此就得熟悉整个项目的具体代码结构及用途

目标是“六亲不认”，让最后成果没人认识

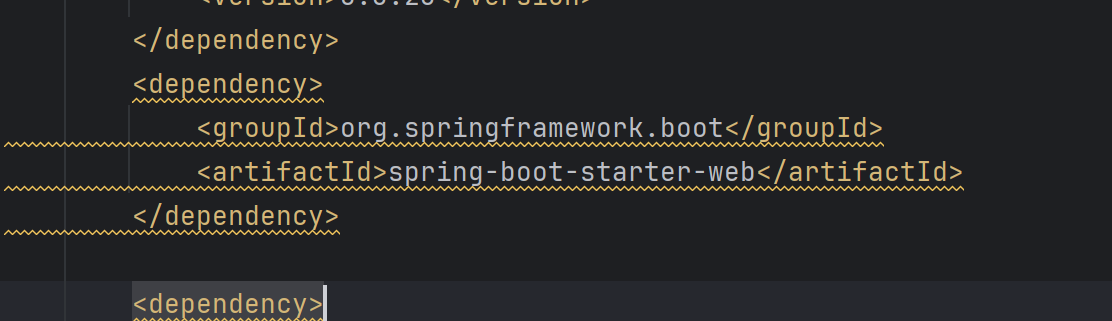
参考教程：https://www.bilibili.com/video/BV1rmkqY9Evz/?spm\_id\_from=333.337.search-card.all.click&vd\_source=033210b7dc9c27e8658b920876ca3033

## 项目介绍

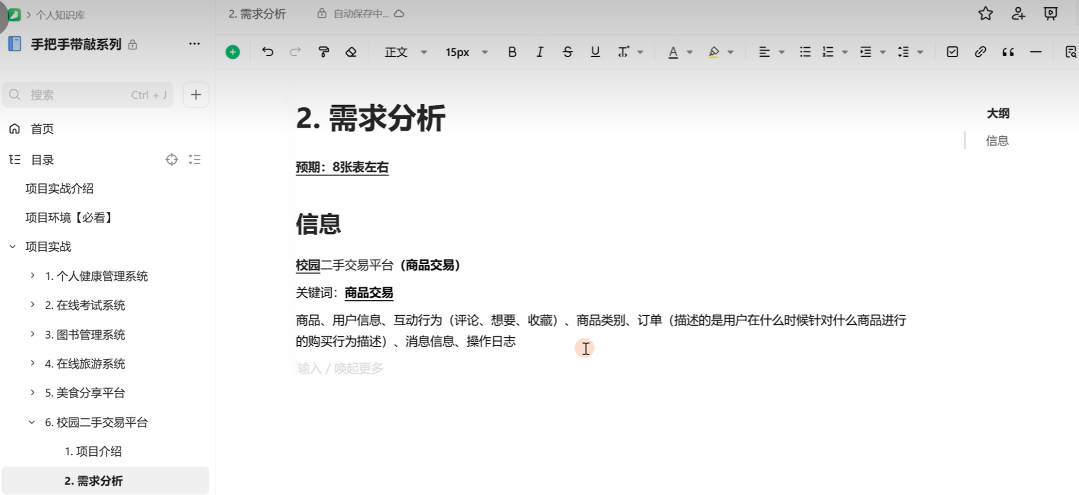


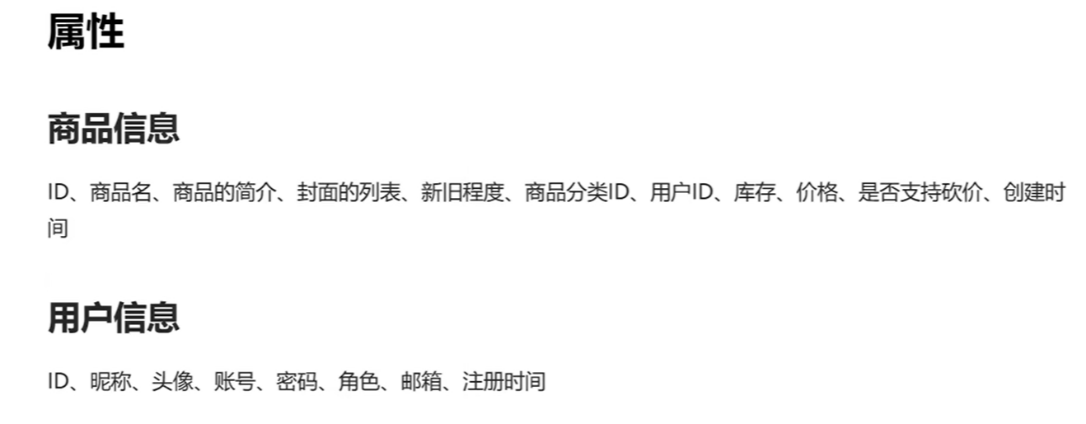
[Spring MVC 与 Spring Boot：如何让你的 Web 应用既灵活又高效？](https://bbs.huaweicloud.com/blogs/447695)

添加了 spring-boot-starter-web 依赖，Spring Boot 就会自动配置并启动一个支持 Spring MVC 的 Web 应用环境。



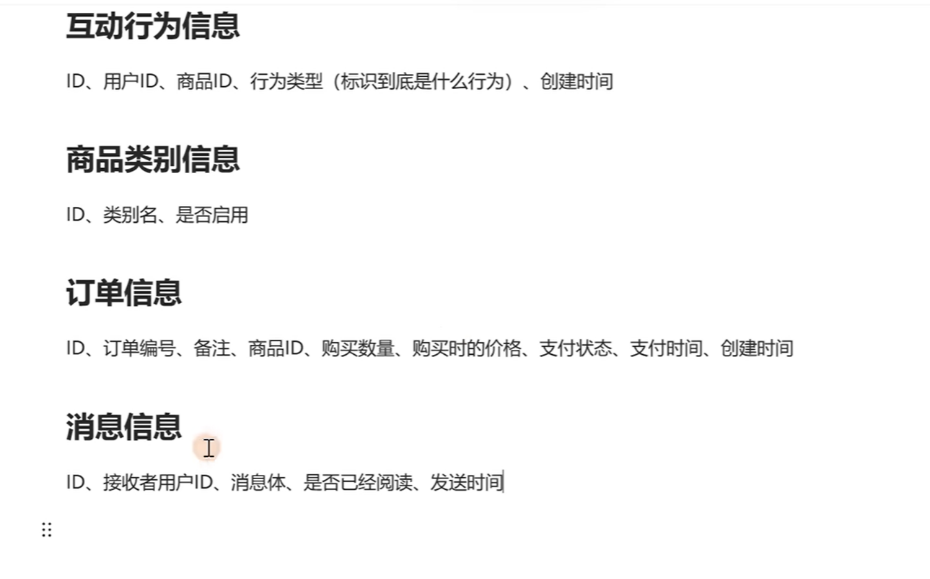
## 需求分析

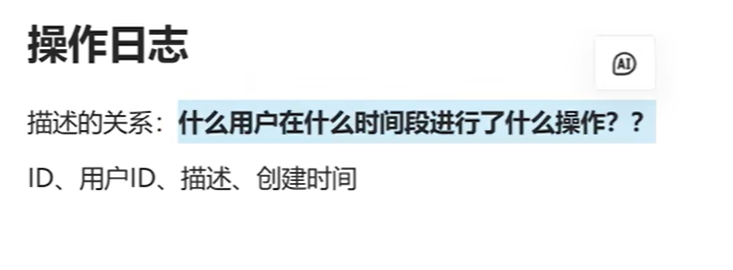








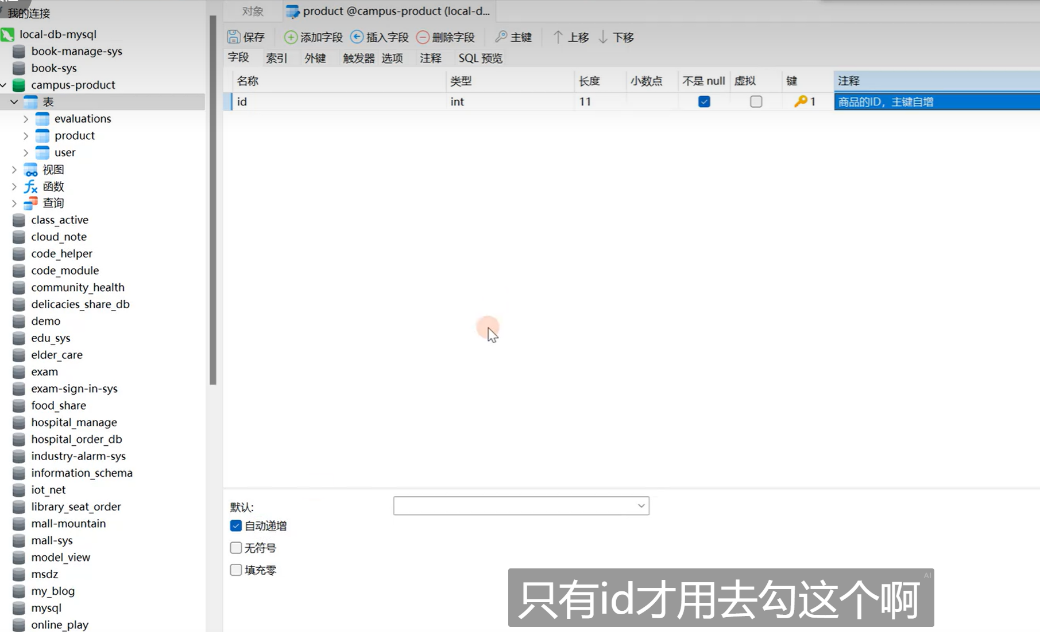




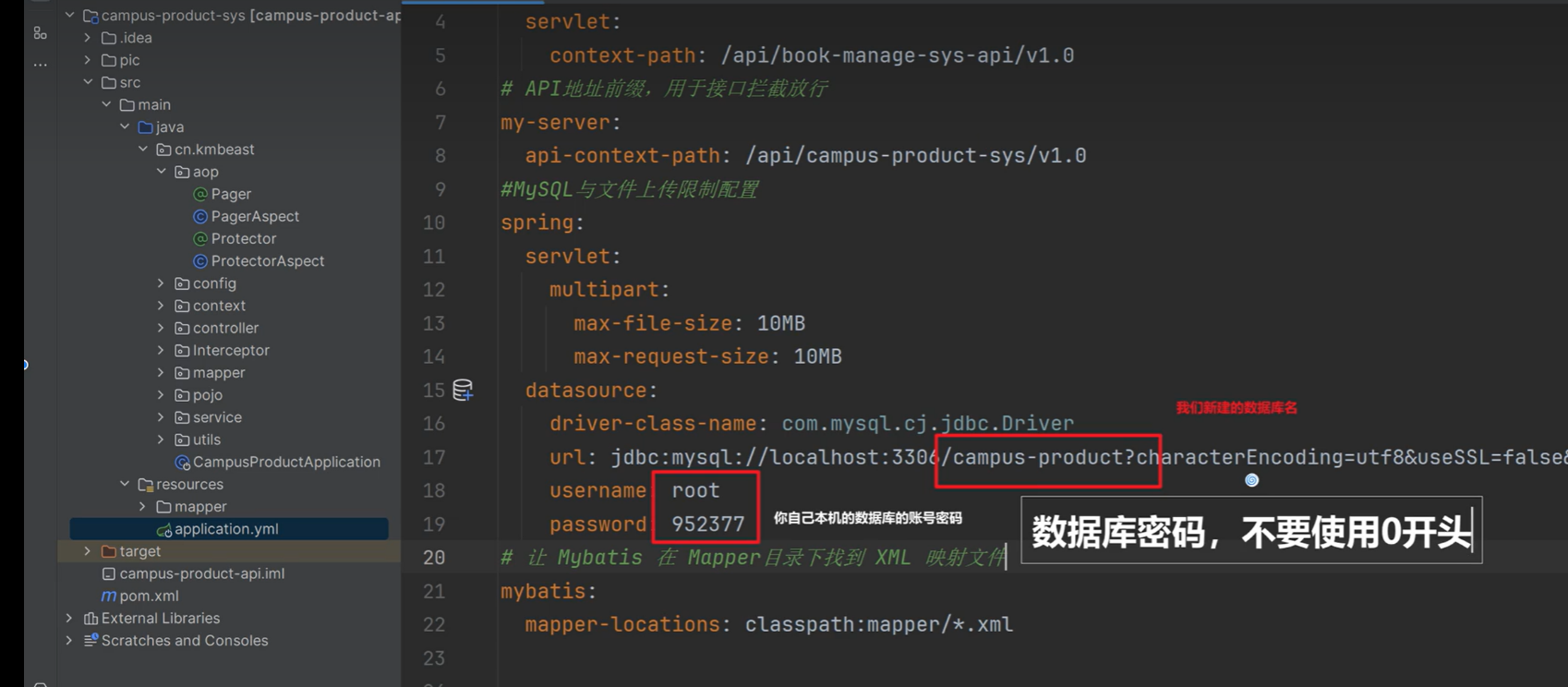
## 数据库设计

一开始就该跟着这个教程配置环境，一步步很详细，前面都做过了，大坑小坑也踩过，直接往下

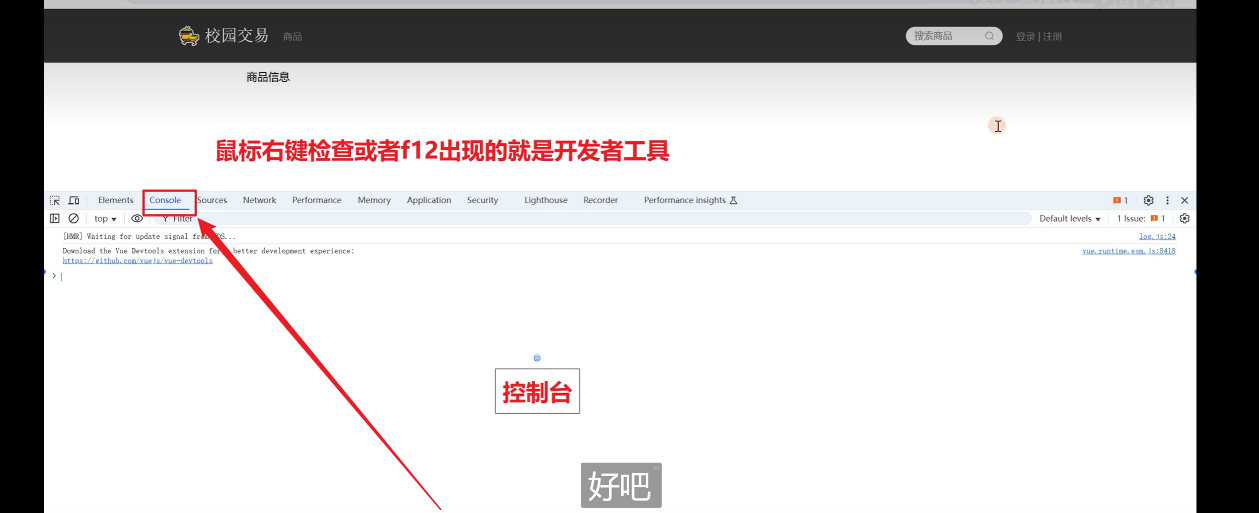
视频教程文档：https://www.yuque.com/yuqueyonghutqcgfo/ex2txh/bxv1aolgy9kb2wtz?#oRuuK（感谢up主，非常好的教程）

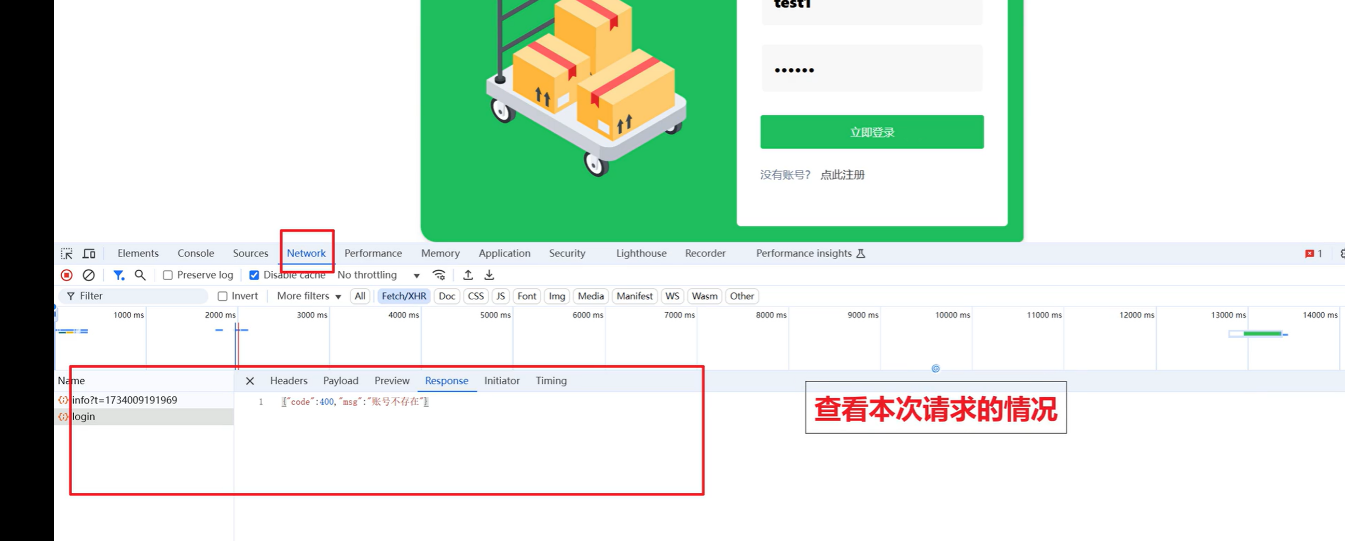


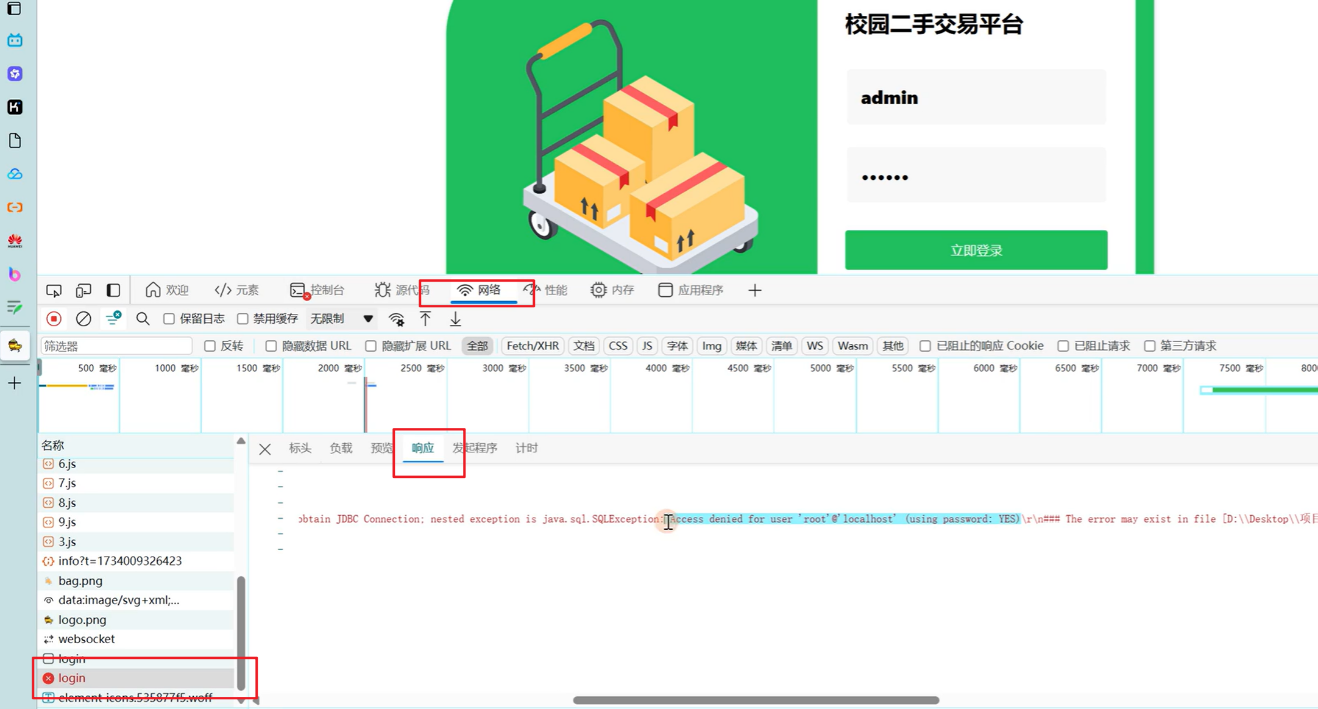
这里up主导入了脚手架，https://blog.csdn.net/zw45607076875/article/details/142573075 即基本架构



但凡早点看我就不至于登录不上了。。。相见恨晚



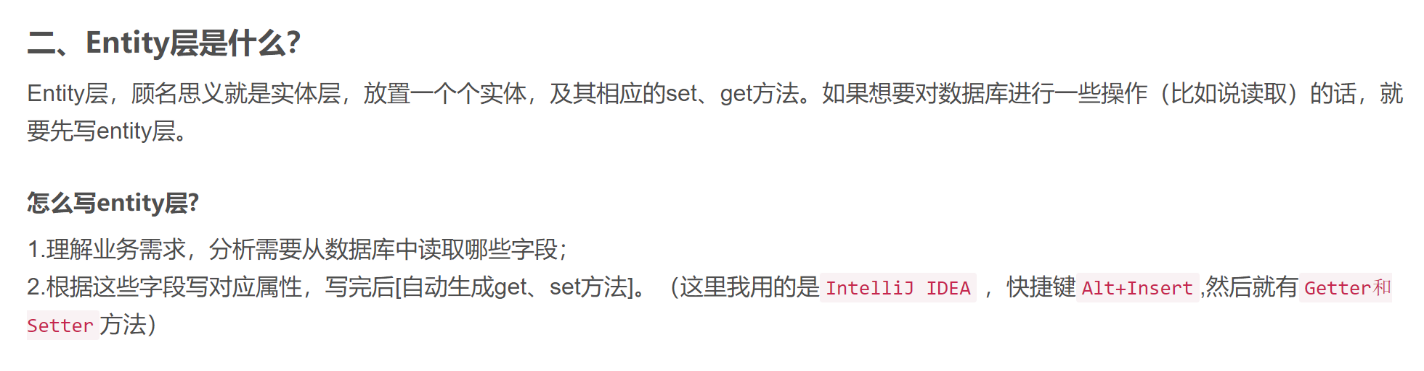


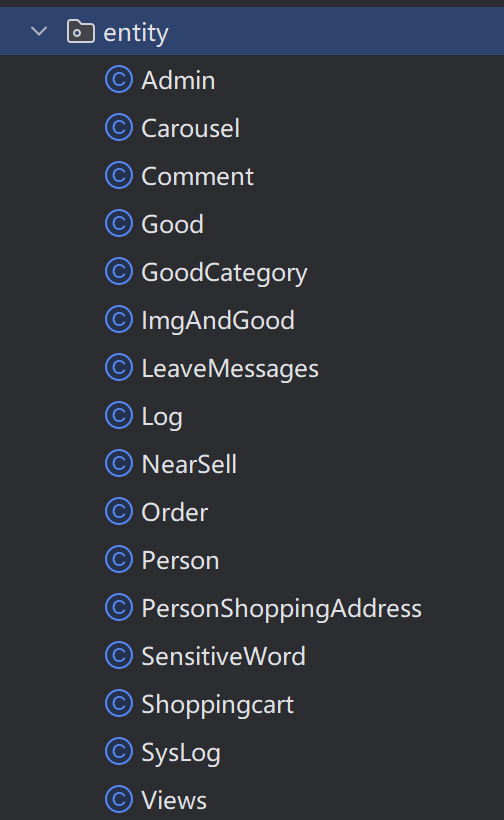


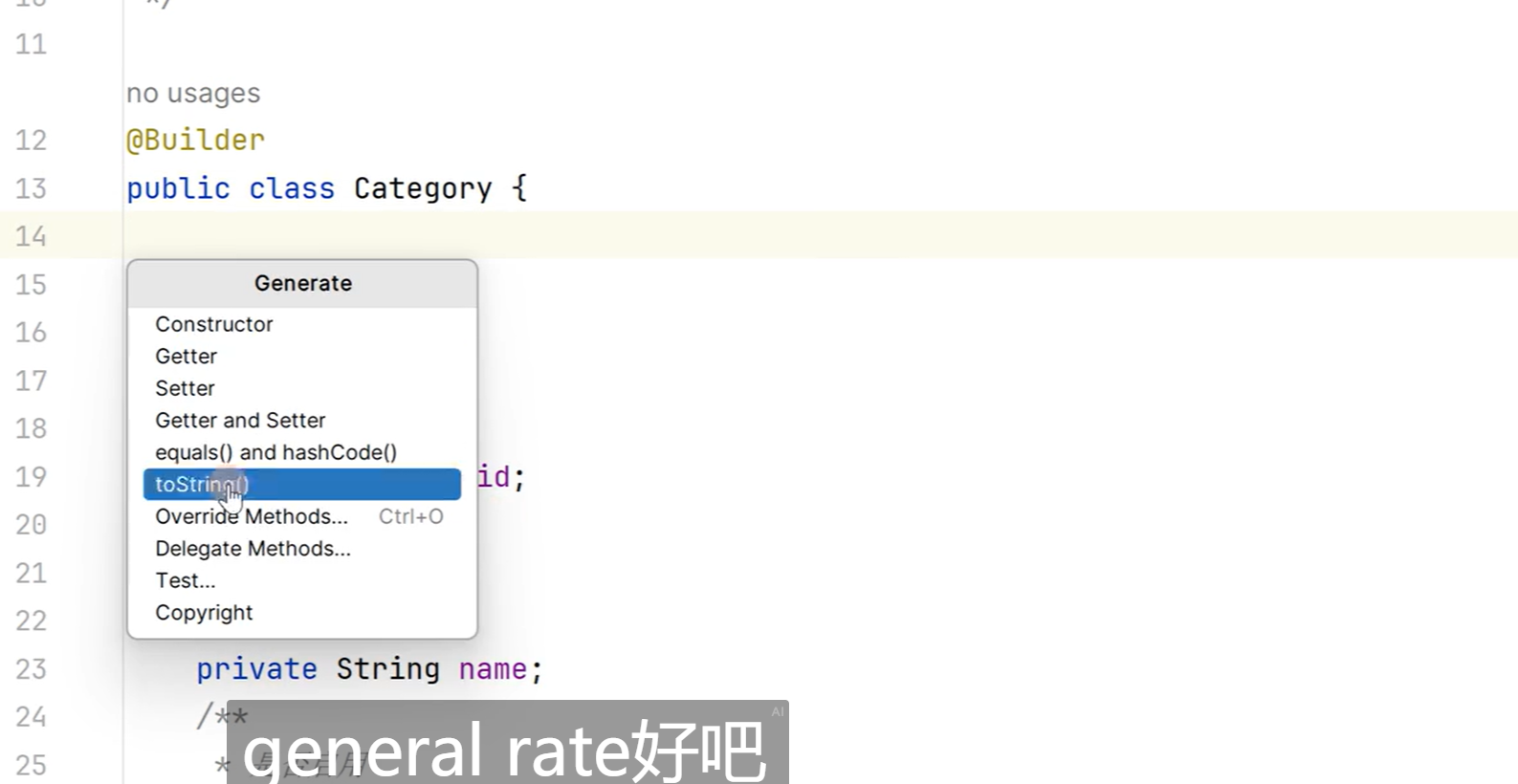
## 实体类建设

数据库里的每张表都要有相应的实体类去对应，在后端的entity文件夹中

entity层





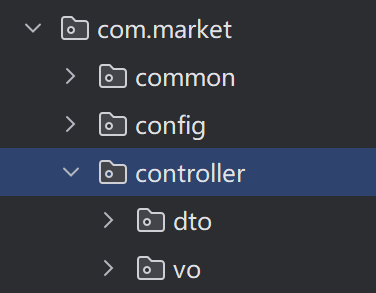


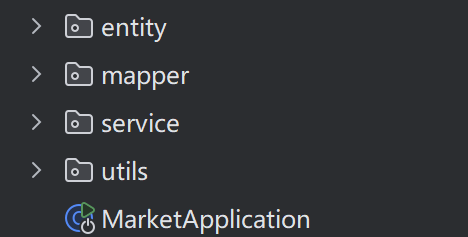
## 商品类别管理

这节课涉及到是control层、dao层等概念，先了解一下

Spring Boot工程项目目录的基本结构





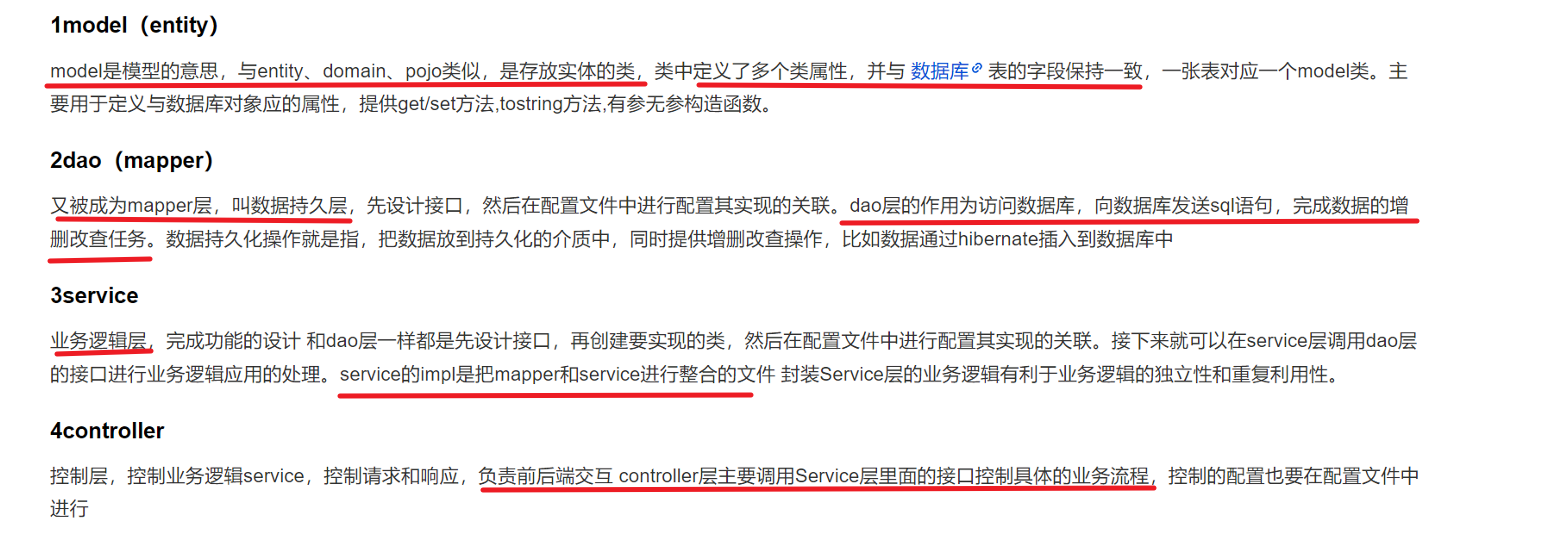


相关架构的概念 <https://blog.csdn.net/Restarting2019/article/details/122296373>

<https://blog.csdn.net/u011095110/article/details/86088976>

<https://cloud.tencent.com/developer/article/1921913>

这个写得全面些



这个涵盖得好，视频中的pojo也明白了是model（entity）

dao（Data Access Object）是mapper层，跟数据库打交道

接着看看Ai的大白话

1. controller（控制器）

负责接收用户的请求（比如浏览器或 App 发来的 HTTP 请求），调用其他模块处理，最后返回响应。

例子：用户访问 /user/profile，UserController 会接收这个请求，调用 UserService 查询用户信息，再返回给前端。

底下还有dto层和vo层

在 controller 层中，DTO（Data Transfer Object）和 VO（View Object）都是用来封装和传递数据的对象

1. DTO（Data Transfer Object，数据传输对象）

作用：

接收前端请求的数据（比如表单提交、API参数）。

隔离实体类（entity），避免直接暴露数据库字段（比如敏感字段 password）。

将多个参数封装成一个对象，方便传递（比如注册时需要 username + password + email）。

2. VO（View Object，视图对象）

作用：

返回给前端的数据（比如接口响应）。

对原始数据（如 entity 或数据库查询结果）加工处理，隐藏敏感信息，或者组合多个数据源。

适配前端展示需求（比如日期格式化、字段重命名）。

总结

DTO 是“前端 → 后端”的数据：负责接收请求参数。

VO 是“后端 → 前端”的数据：负责返回处理后的结果。

2. entity（实体）

是什么：对应数据库中的表结构，一个实体类代表一张表，类的属性对应表的字段。

例子：

数据库有 user 表，字段是 id, name, age，对应的实体类 User 就会有这三个属性。

3. mapper（映射器）

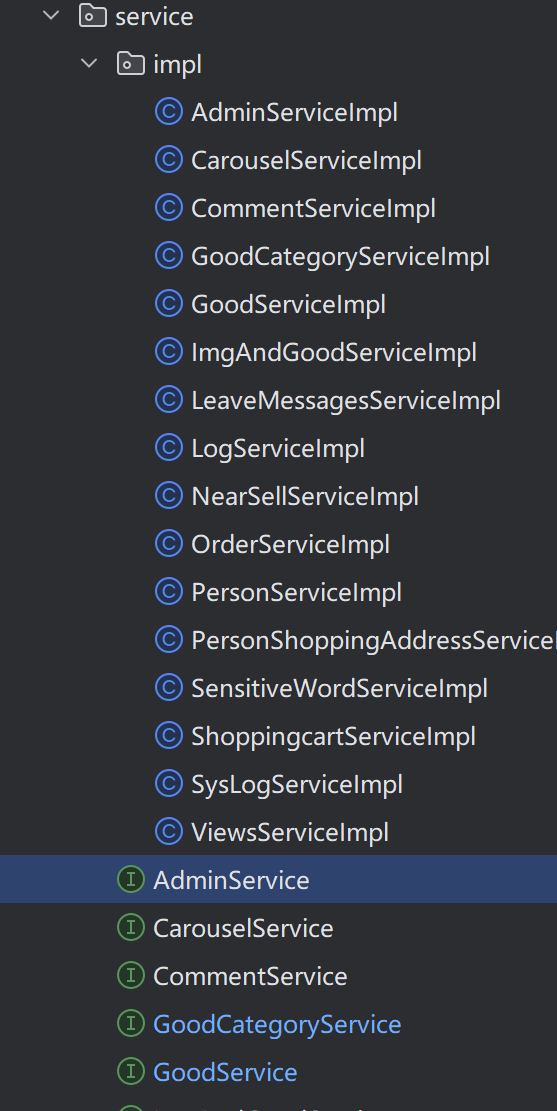
是什么：负责操作数据库，将 Java 对象和数据库表进行转换（比如增删改查）。

例子：UserMapper 接口中定义了一个方法 findUserById(Long id)，实际会执行 SQL：SELECT \* FROM user WHERE id = ?

4. service（服务层）

是什么：处理具体的业务逻辑（比如订单支付、用户注册），会调用 mapper 操作数据库。

例子：UserService 中有一个方法 registerUser，内部会检查用户名是否重复，再通过 UserMapper 将用户数据存入数据库。



里面还有extends继承和impl层（implement接口的具体实现类）

5. utils（工具类）

是什么：提供通用的工具方法，比如生成随机数、加密密码等。

例子：StringUtils 类中的 isEmpty 方法，判断字符串是否为空。

6. common（通用模块）

是什么：存放项目中多个地方都会用到的公共代码，比如工具类、通用配置、错误处理模板等。

例子：

处理日期格式的类 DateUtils

自定义的全局异常处理类 GlobalExceptionHandler

7. config（配置模块）

是什么：用来存放项目的配置信息，比如数据库连接、第三方 API 的密钥等。

例子：

配置数据库用户名密码的类 DatabaseConfig

配置 Swagger（API 文档工具）的类 SwaggerConfig

总结

这些模块让代码分工明确：

controller 管“接活”，service 管“干活”，mapper 管“搬砖（数据）”，entity 是“砖块”，config/utils/common 是“工具箱”。  
这种结构让代码易读、易扩展，团队协作也更方便。

课程中提到了新插件MybatisX

MybatisX 提供了便捷的 XML 映射文件与 Java 接口之间的跳转功能，让开发者能够快速地在两者之间切换，提高开发效率。

接着是前端的基本结构

<https://www.runoob.com/vue2/vue-directory-structure.html> vue菜鸟教程足够了

<https://blog.csdn.net/qq_42078712/article/details/133950729>

myapp/

├── public/ # 公共资源目录

│ ├── index.html # 项目的入口 HTML 文件

│ └── favicon.ico # 网站的图标

│

├── src/ # 项目源代码目录

│ ├── api/ # 定义和管理向后端服务器发送请求的模块或服务的目录

│ ├── assets/ # 静态资源目录

│ ├── components/ # 存放组件目录

│ ├── router/ # 路由配置目录

│ ├── services/ # 同api/一样功能

│ ├── store/ # 全局状态管理目录

│ ├── styles/ # 同assets/一样功能

│ ├── views/ # 页面组件目录

│ ├── App.vue # 根组件

│ └── main.js # 项目入口文件

│

├── .eslintrc.js # ESLint 配置文件

├── babel.config.js # Babel 配置文件

├── package.json # 项目配置文件

└── README.md # 项目说明文件

虽然但是，没有代码基础果然还是很难看得懂写代码的过程，唉。。。

总结大概的实现步骤

src/main/java

├── com.example.demo

│ ├── controller

│ │ └── CategoryController.java # 控制器

│ ├── pojo/extend

│ │ └── CategoryQueryDto.java # 查询参数类

│ ├── service

│ │ ├── CategoryService.java # 业务接口

│ │ └── impl

│ │ └── CategoryServiceImpl.java # 接口实现

│ └── mapper

│ └── CategoryMapper.java # MyBatis Mapper接口

src/main/resources

├── mapper

│ └── CategoryMapper.xml # SQL映射文件

1.实体类

在pojo//extend包下，新建CategoryQueryDto.java商品类别查询参数类

2.控制器层(controller)

在controller包下，新建CategoryController.javai商品类别控制器

3.业务逻辑层(service)

在service包下，新建CategoryService.java业务逻辑接口 （接口定义）

在service/impl包下，新建CategoryServicelmpl,.java商品类别业务逻辑接口实现类（接口实现）

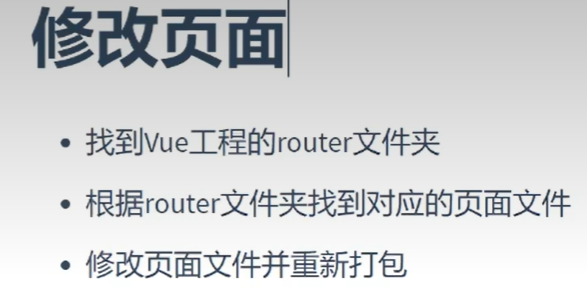
4.数据持久层

在mapper包下，新建CategoryMapper.java持久化接口

5.在resources/mapper包下，新建CategoryMapper.xml文件（SQL映射文件）

没有基础看起来还是很困难，开始剑走偏锋了

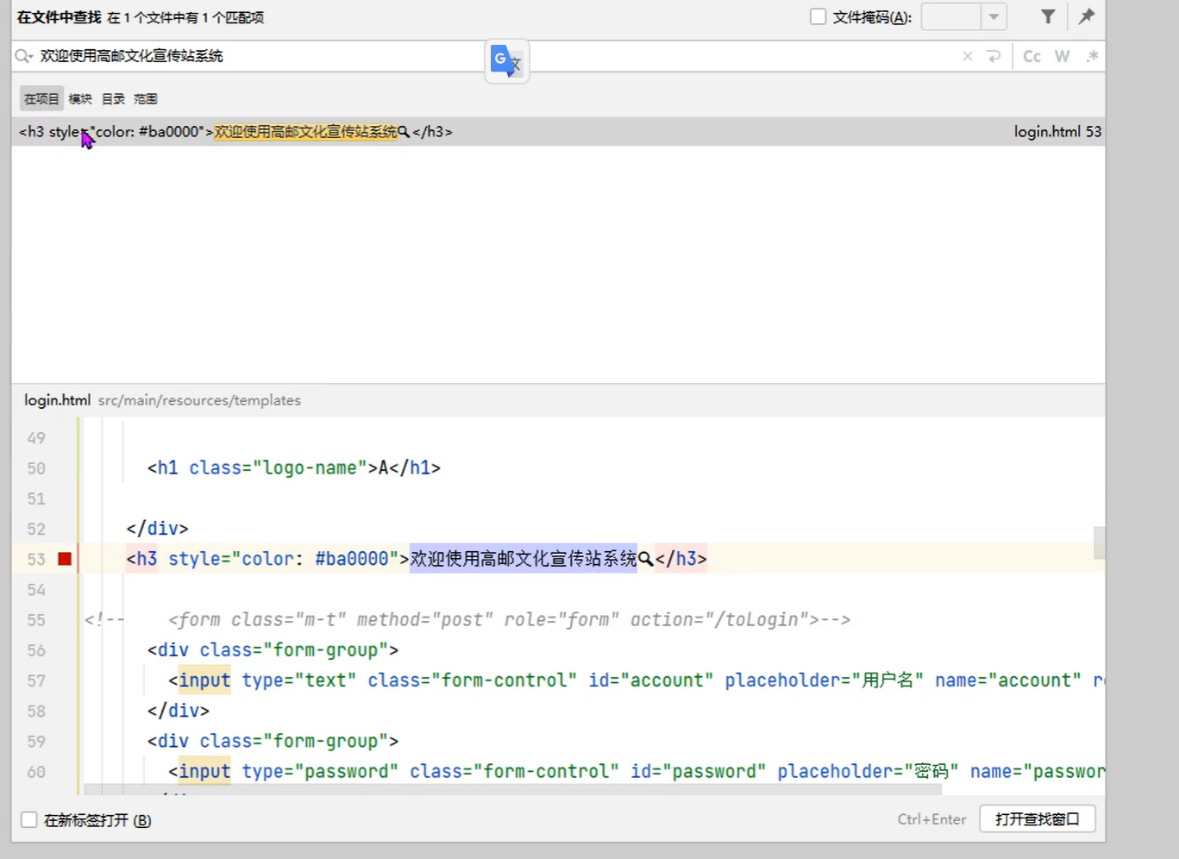
[【小白向】教你如何修改SpringBoot+Vue毕设管理系统的页面](https://www.bilibili.com/video/BV1NZ421J75L/?spm_id_from=333.337.search-card.all.click&vd_source=033210b7dc9c27e8658b920876ca3033)



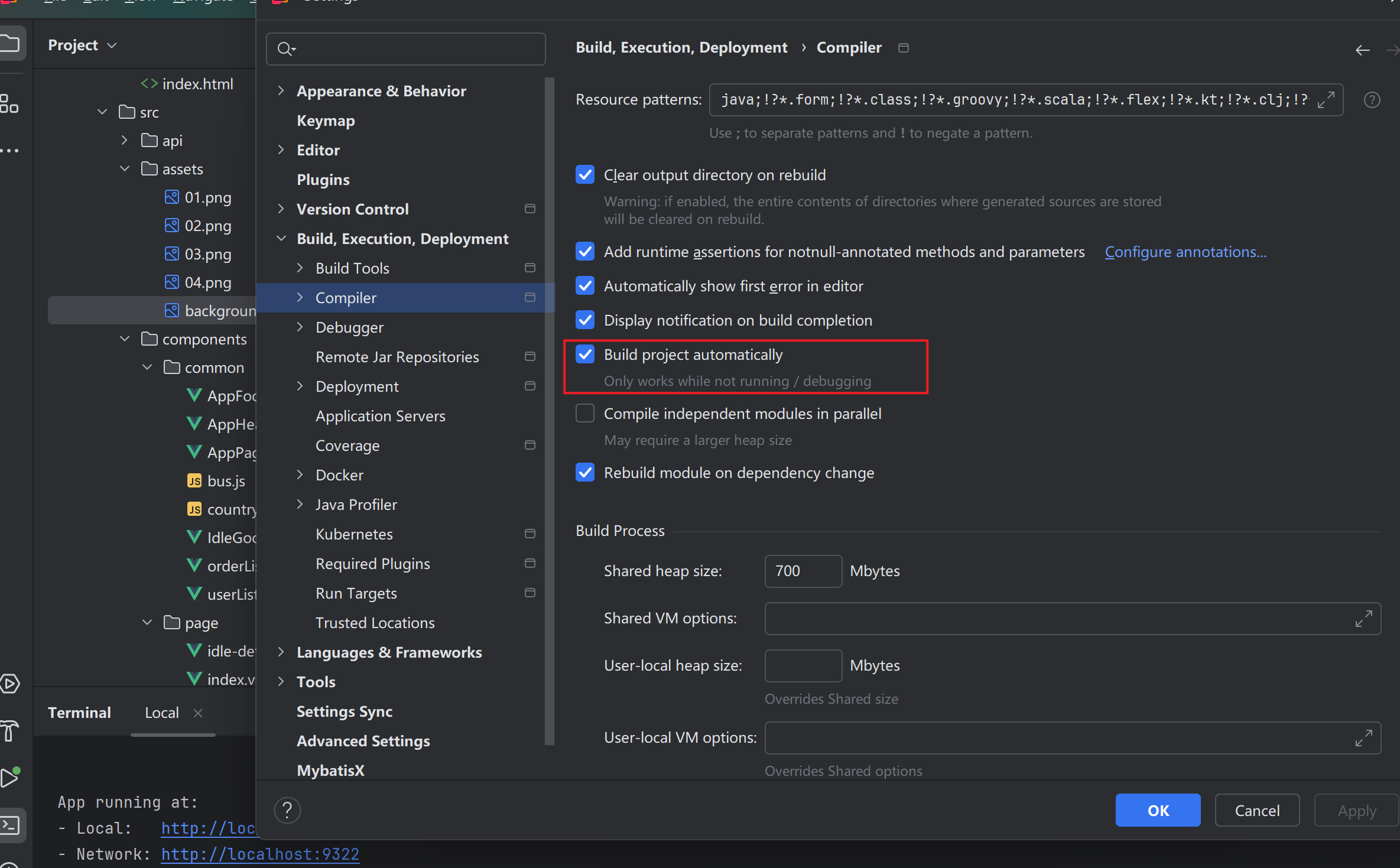
实现单个校园里的线上校园二手物品交易信息管理系统

[如何修改项目的页面](https://www.bilibili.com/video/BV1JF5uzLEms/?spm_id_from=333.337.search-card.all.click&vd_source=033210b7dc9c27e8658b920876ca3033)

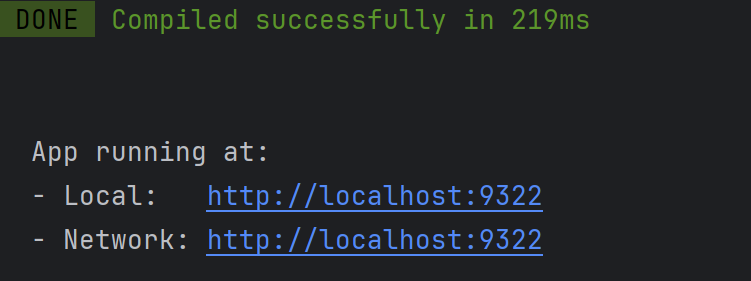
快捷键全局搜索 ctrl+shift+F[（idea 的全局搜索快捷键失效问题）](https://blog.csdn.net/ly_6699/article/details/107795509)



接触到了一个叫热部署的名词



网上有Jrebel插件等各种方法，但是上面IDEA自带的选项配置后就很方便了，修改某个地方之后就能自动编译





3.        DEMO系统演示重点考察以下要点：

（1）        系统整理的功能合理性，界面友好性，以及演示操作时的系统流畅性

（2）        实际系统的数据库设计合理性（展示所设计的数据库表及各表结构）

（3）        系统功能实现的逻辑性（展示部分SQL语句，并简要阐述查询过程）

（4）        其它验收老师提出的相关问题。

* 1. 前端部分 (UserLogin.vue中的注册逻辑)：

market-vue\src\views\Login\UserLogin.vue

Apply

// 用户点击注册按钮  
personRegister() {  
 this.$axios({  
 url: this.$http + "/person/personRegister",  
 method: 'post',  
 data: this.formX, // 包含用户填写的注册信息  
 headers: {'Content-Type': 'application/json'}  
 }).then(res => {  
 // 处理注册结果...  
 })  
}

* 1. 后端Controller (PersonController.java)：

market\src\main\java\com\market\controller\PersonController.java

Apply

// 用户注册接口  
@PostMapping("/personRegister")  
public Result personRegister(@RequestBody Person person, HttpServletRequest request) {  
 // 生成用户ID  
 Integer maxId = personService.getMaxId();  
 person.setPersonId(maxId + 1);  
   
 // 设置注册时间和创建时间  
 person.setRegisterDate(new Date());  
 persoe; // 创建时间

注意：实际代码中密码应该进行加密存储，但当前代码中未看到加密处理，建议补充MD5或BCrypt加密逻辑

n.setCreateTime(new Date());  
   
 // 保存到数据库  
 personService.saveOrUpdate(person);  
 return Result.success();  
}

* 1. Service层 (PersonService.java)：

market\src\main\java\com\market\service\PersonService.java

Apply

public interface PersonService extends IService<Person> {  
 // 继承MyBatis-Plus的saveOrUpdate方法  
 Integer getMaxId(); // 获取当前最大ID  
}

* 1. Mapper层 (PersonMapper.xml)：

market\src\main\resources\mapper\PersonMapper.xml

Apply

<select id="getMaxId" resultType="java.lang.Integer">  
 SELECT MAX(person\_id) FROM person  
</select>

关键流程说明：

* 1. 前端收集用户输入的账号、密码、姓名、性别、学校等信息
  2. 通过POST请求将数据发送到/person/personRegister接口
  3. 后端生成连续的用户ID（通过查询当前最大ID+1）
  4. 自动设置注册时间和创建时间为当前时间
  5. 使用MyBatis-Plus的saveOrUpdate方法将数据持久化到person表

数据库表结构（对应Person.java实体类）：

java

Apply

@TableId(value = "person\_id", type = IdType.INPUT)  
private Integer personId; // 用户ID  
private String account; // 账户  
private String userName; // 用户名   
private String password; // 密码  
private Integer sex; // 性别  
private Integer campusId; // 学校ID  
private Date registerDate; // 注册时间  
private String avatarUrl; // 头像URL  
private Date createTim

订单的实现

根据代码库中的实现，订单信息从前端到数据库的完整流程如下：

* 1. 前端提交订单（GoodInformation.vue）：

market-vue\src\components\user\GoodInformation.vue

Apply

SubmitOrder() {  
 this.$axios({  
 url: this.$http + '/order/createOrder',  
 method: 'post',  
 data: {  
 goodId: this.goodId,  
 count: this.count,  
 customerId: this.currentUserId,  
 totalPrice: this.totalPrice  
 },  
 headers: {token: localStorage.getItem("token")}  
 }).then(res => {  
 // 处理订单创建结果...  
 })  
}

* 1. 后端Controller处理（OrderController.java）：

market\src\main\java\com\market\controller\OrderController.java

Apply

@PostMapping("/createOrder")  
public Result createOrder(@RequestBody OrderDTO orderDTO, HttpServletRequest request) {  
 String token = request.getHeader("token");  
 if (TokenUtil.checkToken(token)) {  
 // 生成订单号  
 Integer maxId = orderService.getMaxId();  
 Order order = new Order();  
 order.setOrderId(maxId + 1);  
   
 // 设置订单信息  
 order.setGoodId(orderDTO.getGoodId());  
 order.setCustomerId(orderDTO.getCustomerId());  
 order.setCount(orderDTO.getCount());  
 order.setTotalPrice(orderDTO.getTotalPrice());  
 order.setCreateTime(new Date());  
   
 // 保存订单  
 orderService.save(order);  
 return Result.success();  
 }  
 return Result.fail();  
}

* 1. Service层处理（OrderService.java）：

market\src\main\java\com\market\service\OrderService.java

Apply

public interface OrderService extends IService<Order> {  
 // 继承MyBatis-Plus的save方法  
 Integer getMaxId(); // 获取当前最大订单ID  
}

* 1. 数据库操作（OrderMapper.xml）：

market\src\main\resources\mapper\OrderMapper.xml

Apply

<insert id="insertOrder" parameterType="com.market.entity.Order">  
 INSERT INTO `order`   
 (order\_id, good\_id, customer\_id, count, total\_price, create\_time)  
 VALUES   
 (#{orderId}, #{goodId}, #{customerId}, #{count}, #{totalPrice}, #{createTime})  
</insert>

完整流程说明：

* 1. 用户在商品详情页点击"立即购买"触发SubmitOrder方法
  2. 前端收集商品ID、购买数量、用户ID、总价等信息
  3. 通过POST请求发送到/order/createOrder接口
  4. 后端生成连续订单ID（当前最大ID+1）
  5. 设置订单创建时间为当前时间
  6. 调用MyBatis-Plus的save方法持久化到order表

数据库表结构（对应Order.java实体类）：

java

Apply

@TableId(value = "order\_id", type = IdType.INPUT)  
private Integer orderId; // 订单ID  
private Integer goodId; // 商品ID  
private Integer customerId; // 买家ID  
private Integer count; // 购买数量  
private BigDecimal totalPrice;// 总金额  
private Integer orderStatus; // 订单状态  
private Date payTime; // 支付时间  
private String payWays; // 支付方式  
private Date createTime; // 创建时间

关键补充逻辑：

* 1. 库存校验：在GoodService中实现库存扣减

market\src\main\java\com\market\service\impl\GoodServiceImpl.java

Apply

public boolean reduceStock(Integer goodId, Integer count) {  
 Good good = getById(goodId);  
 if(good.getCounts() >= count) {  
 good.setCounts(good.getCounts() - count);  
 return updateById(good);  
 }  
 return false;  
}

* 1. 支付状态更新：支付成功后调用支付接口

market\src\main\java\com\market\controller\OrderController.java

Apply

@PostMapping("/payOrder")  
public Result payOrder(@RequestBody OrderPayWayDTO payDTO) {  
 Order order = orderService.getById(payDTO.getOrderId());  
 order.setPayTime(new Date());  
 order.setPayWays(payDTO.getPayWay());  
 order.setOrderStatus(1); // 标记为已支付  
 return Result.success(orderService.updateById(order));  
}

该实现中需要注意：

* 1. 需要添加事务管理保证库存扣减和订单创建的原子性
  2. 实际支付需要集成第三方支付接口（当前代码仅为状态标记）
  3. 订单状态需要定义明确的枚举值（当前使用1表示已支付）

猜你喜欢怎么做到的

基于商品分类的简单推荐（需要结合category\_id字段扩展）

